Lab Manual

for

Linear Algebra

by

Jim Hefferon

Cover: my Chocolate Lab, Suzy.

Preface

This collection supplements the text *Linear Algebra*[1] with a number of explorations that help students solidify and extend their understanding of the subject, using the mathematical software Sage.[2]

Naturally the text presents its material using examples and practice problems that are small-sized and have manageable numbers: an assignment to multiply a pair of three by three matrices of small integers will build intuition, whereas asking students to do that same by-hand question with twenty by twenty matrices of ten decimal place numbers would be badgering. (And, even more worrisome, having students focus their intellectual energy on calculations instead of directing their attention to the ideas and proofs misleads them as to what the subject is about.)

However, mathematical software can mitigate this by extending the reach of what is reasonable to bigger systems, harder numbers, and computations that — while too much to do by hand — yield interesting information when they are done by a machine. For instance, an advantage of learning how to handle these tougher computations is that they are more like the ones that appear when students apply Linear Algebra to other subjects. Another advantage is that students see new ideas such as runtime growth measures.

Well then, why not teach straight from the computer system?

A major goal of any undergraduate Mathematics program is to over time move students toward a higher-level, more abstract, grasp of the subject. For instance, Calculus classes work on elaborate computations while later courses spend more effort on concepts and proofs, leaving less for the details of calculations. The text *Linear Algebra* fits into this development process. To develop a higher-level understanding of the material we want to keep the focus on vector spaces and linear maps. In the text's exposition the computations are a way to develop that understanding, not the main point. Some instructors may find that for their students this work is best left aside altogether, keeping a tight focus on the core material, while other instructors have students who will benefit from the increased reach that the software provides. This manual gives teachers the freedom to make the choice that suits their class.

Why Sage?

Sage is a very powerful mathematical software systems but so are many others. This manual uses it because it is Free[3] and Open Source[4] software.

[1]The text's home page http://joshua.smcvt.edu/linearalgebra has the PDF, the ancillary materials, and the LaTeX source. [2]See http://www.sagemath.org for the software and documentation. [3]The Free Software Foundation page http://www.gnu.org/philosophy/free-sw.html gives background and a definition. [4]See http://opensource.org/osd.html for a definition.

In *Open Source Mathematical Software* [?][1] the authors argue that for Mathematics the best way forward is to use software that is Open Source.

> Suppose Jane is a well-known mathematician who announces she has proved a theorem. We probably will believe her, but she knows that she will be required to produce a proof if requested. However, suppose now Jane says a theorem is true based partly on the results of software. The closest we can reasonably hope to get to a rigorous proof (without new ideas) is the open inspection and ability to use all the computer code on which the result depends. If the program is proprietary, this is not possible. We have every right to be distrustful, not only due to a vague distrust of computers but because even the best programmers regularly make mistakes.
>
> If one reads the proof of Jane's theorem in hopes of extending her ideas or applying them in a new context, it is limiting to not have access to the inner workings of the software on which Jane's result builds.

Professionals choose their tools by balancing many factors but this argument is persuasive. This manual uses Sage because it is very capable, including at Linear Algebra, because students can learn a great deal from it, and because it is Free.

This manual

This is Free. Get the latest version from http://joshua.smcvt.edu/linearalgebra. Also see that page for the license details and for the LaTeX source, including this manual.

I am glad to hear suggestions or corrections, especially from instructors who have class-tested the material. My contact information is on the same page.

The Sage output in this manual was generated automatically so it is sure to be accurate, except that I have (automatically) edited a few lines for length. My Sage identifies itself in this way.

```
'Sage Version 5.2, Release Date: 2012-07-25'
```

Acknowledgements

I am glad for this chance to thank the Sage Development Team for their work. In particular, without [?] this manual would not have happened. I am glad also for the chance to mention [?] as an inspiration.

Jim Hefferon
Mathematics, Saint Michael's College
Colchester, Vermont USA
2012-Sep-10

[1] See http://www.ams.org/notices/200710/tx071001279p.pdf for the full text.

Contents

Geometry of Linear Maps

Sage can illustrate the geometric effect of linear maps. Here we focus on transformations of the plane $\mathbb{R}^2$.

Lines map to lines

The pictures in this chapter are based on the observation that linear maps send lines to lines.

Consider a domain space $\mathbb{R}^d$ and codomain space $\mathbb{R}^c$, along with the linear map h. We get a line in the domain by fixing a vector of slopes $\vec{m} \in \mathbb{R}^d$ and a vector of offsets from the origin $\vec{b}$ and taking the set $\ell = \{\vec{v} = \vec{m} \cdot s + \vec{b} \bigm| s \in \mathbb{R}\}$. Then the image of this line $h(\ell)$ is the set $\{h(\vec{m}s + \vec{b}) \bigm| s \in \mathbb{R}\} = \{h(\vec{m})s + h(\vec{b}) \bigm| s \in \mathbb{R}\}$. This is a line in the codomain $\mathbb{R}^c$ with the vector of slopes $h(\vec{m})$ and the vector of offsets $h(\vec{b})$.

For example, consider the transformation $t\colon \mathbb{R}^2 \to \mathbb{R}^2$ that rotates vectors counterclockwise by $\pi/6$

$$\mathrm{Rep}_{E_2,E_2}(t) = \begin{pmatrix} \cos(\pi/6) & -\sin(\pi/6) \\ \sin(\pi/6) & \cos(\pi/6) \end{pmatrix} = \begin{pmatrix} \sqrt{3}/2 & -1/2 \\ 1/2 & \sqrt{3}/2 \end{pmatrix}$$

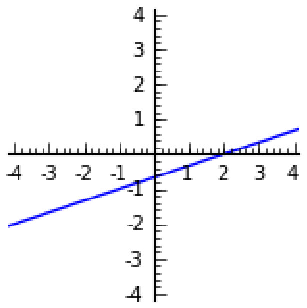
(remember that this differs from the matrix given in the book because Sage has vectors multiply from the left). The plane line $y = 3x + 2$ is this set.

$$\ell = \{\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \end{pmatrix} \cdot s + \begin{pmatrix} 2 \\ 0 \end{pmatrix} \bigm| s \in \mathbb{R}\}$$

The rotated line is this set.

$$h(\ell) = \{\begin{pmatrix} x \\ y \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 3\sqrt{3}+1 \\ -3+\sqrt{3} \end{pmatrix} \cdot s + \begin{pmatrix} \sqrt{3} \\ 1 \end{pmatrix} \bigm| s \in \mathbb{R}\}$$

```
sage: s = var('s')
sage: ell = parametric_plot((3*s+2,1*s), (s, -10, 10))
sage: ell.set_axes_range(-4, 4, -4, 4)
sage: ell.save("sageoutput/plot_action0.png", figsize=2.5, fontsize=7)
```



So the routine finds the effect of the map

$$\begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

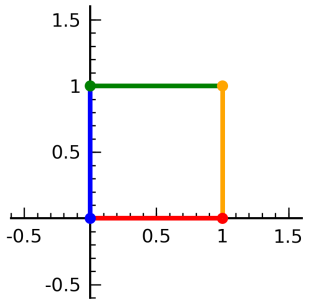
on the four corners of the square

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \overset{t}{\longmapsto} \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 0 \end{pmatrix} \overset{t}{\longmapsto} \begin{pmatrix} a \\ b \end{pmatrix} \quad \begin{pmatrix} 1 \\ 1 \end{pmatrix} \overset{t}{\longmapsto} \begin{pmatrix} a+c \\ b+d \end{pmatrix} \quad \begin{pmatrix} 0 \\ 1 \end{pmatrix} \overset{t}{\longmapsto} \begin{pmatrix} c \\ d \end{pmatrix}$$

and plots four line segments.

The unit square

Consider a linear transformation $t \colon \mathbb{R}^2 \to \mathbb{R}^2$ applied to this unit square resting in the first quadrant.



This code generates that picture.

```
sage: load "plot_action.sage"
sage: p = plot_square_action(1,0,0,1)
sage: p.set_axes_range(-0.5, 1.5, -0.5, 1.5)
sage: p.save("sageoutput/plot_action1.png")
```

The routine `plot_square_action(a, b, c, d)` (whose code is at the end of this chapter) shows the effect of the matrix

$$\text{Rep}_{E_2,E_2}(t) = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

on a unit square. The code above gives this routine the identity matrix, so it plots the square unchanged.

This routine is based on the observation that linear maps send lines to lines. Consider the effect of the map $h\colon \mathbb{R}^n \to \mathbb{R}^m$ on the line $\ell = \{\vec{v} = \vec{m}\cdot s + \vec{b} \bigm| s \in \mathbb{R}\}$. The image is the set $\{h(\vec{m}s + \vec{b}) \bigm| s \in \mathbb{R}\} = \{h(\vec{m})s + h(\vec{b}) \bigm| s \in \mathbb{R}\}$, which is a line in $\mathbb{R}^m$. So the routine finds the effect of the map

$$\begin{pmatrix} x & y \end{pmatrix}\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

on the four corners of the square

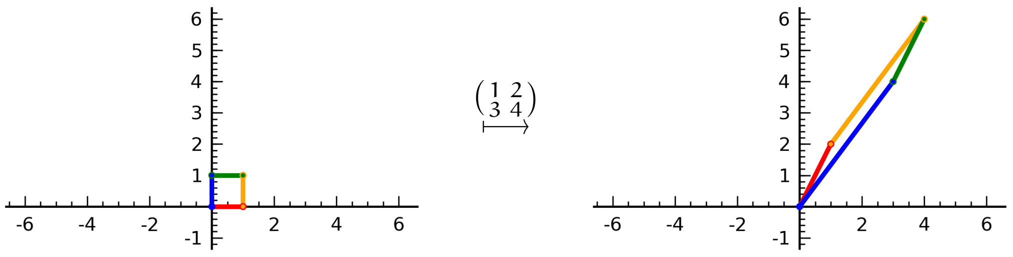
$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \overset{t}{\longmapsto} \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 0 \end{pmatrix} \overset{t}{\longmapsto} \begin{pmatrix} a \\ b \end{pmatrix} \quad \begin{pmatrix} 1 \\ 1 \end{pmatrix} \overset{t}{\longmapsto} \begin{pmatrix} a+c \\ b+d \end{pmatrix} \quad \begin{pmatrix} 0 \\ 1 \end{pmatrix} \overset{t}{\longmapsto} \begin{pmatrix} c \\ d \end{pmatrix}$$

and plots four line segments.

For example, this code

```
sage: load "plot_action.sage"
sage: q = plot_square_action(1,0,0,1)
sage: p = plot_square_action(1,2,3,4)
sage: q.set_axes_range(-6, 6, -1, 6)
sage: p.set_axes_range(-6, 6, -1, 6)
sage: q.save("sageoutput/plot_action2.png")
sage: p.save("sageoutput/plot_action3.png")
```

generates these pictures showing the effect of the matrix.[1]



The colors are there to show that transformations can change orientations. Suppose that we take the natural order of colors to be red, orange, green, and then blue. Then the domain square is a counterclockwise shape, while the transformed square is clockwise.

Transformation acts uniformily?

Recall that any matrix T factors as $H = PBQ$, where P and Q are nonsingular and B is a partial-identity matrix. Recall also that nonsingular matrices factor into elementary matrices $PBQ = T_nT_{n-1}\cdots T_jBT_{j-1}\cdots T_1$, which are matrices that come from the identity I after one Gaussian step

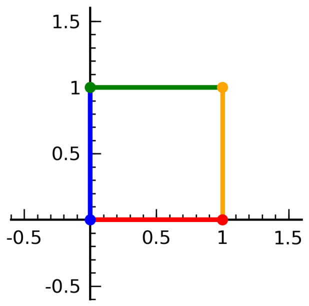
$$I \xrightarrow{k\rho_i} M_i(k) \qquad I \xrightarrow{\rho_i\leftrightarrow\rho_j} P_{i,j} \qquad I \xrightarrow{k\rho_i+\rho_j} C_{i,j}(k)$$

for $i \neq j$, $k \neq 0$. So if we understand the effect of a linear map described by a partial-identity matrix and the effect of the linear maps described by the elementary matrices then we will in some sense understand the effect of any linear map. (To understand them we mean to give a description of their geometric effect; the pictures below stick to transformations of $\mathbb{R}^2$ for ease of drawing but the principles extend for maps from any $\mathbb{R}^n$ to any $\mathbb{R}^m$.)

[1]The remaining examples in this chapter omit the fiddly lines that load, save, set the axis ranges, etc.

Maps preserve lines through the origin

```
# plot_action.sage
# Show the action of a 2x2 matrix on the top half of a unit circle

DOT_SIZE = .02

def color_circle_list(a, b, c, d, colors):
    """Return list of graph instances for the action of a 2x2 matrix on
    half of the unit circle.  That circle is broken into chunks each
    colored a different color.
      a, b, c, d  reals  entries of the matrix
      colors  list of rgb tuples; len of this list is how many chunks
    """
    r = []
    t = var('t')
    n = len(colors)
    for i in range(n):
      color = colors[i]
        x(t) = a*cos(t)+b*sin(t)
        y(t) = c*cos(t)+d*sin(t)
        g = parametric_plot((x(t), y(t)),
                            (t, pi*i/n, pi*(i+1)/n),
                            color = color)
      r.append(g)
        r.append(circle((x(pi*i/n), y(pi*i/n)), DOT_SIZE, color=color))
    r.append(circle((x(pi), y(pi)), 2*DOT_SIZE, color='black',
                    fill = 'true'))
    r.append(circle((x(pi), y(pi)), DOT_SIZE, color='white',
                    fill = 'true'))
    return r

def plot_circle_action(a, b, c, d, n = 12):
    """Show the action of the matrix with entries a, b, c, d on half
    of the unit circle, as the circle and the output curve, broken into
    a number of colors.
     a, b, c, d  reals  Entries are upper left, ur, ll, lr.
     n = 12  positive integer  Number of colors.
    """
    colors = rainbow(n)
    G = Graphics()  # holds graph parts until they are to be shown
    for f_part in color_circle_list(1,0,0,1,colors):
        G += f_part
    for g_part in color_circle_list(a,b,c,d,colors):
        G += g_part
    return plot(G)

THICKNESS = 1.75
ZORDER = 5
def color_square_list(a, b, c, d, colors):
    """Return list of graph instances for the action of a 2x2 matrix
    on a unit square.  That square is broken into sides, each colored a
    different color.
      a, b, c, d  reals  entries of the matrix
      colors  list of rgb tuples; len of this list is at least four
    """
    r = []
    t = var('t')
    # Four sides, ccw around square from origin
    r.append(parametric_plot((a*t, b*t), (t, 0, 1),
                             color = colors[0], zorder=ZORDER, thickness=THICKNESS)
    r.append(parametric_plot((a+c*t, b+d*t), (t, 0, 1),
                             color = colors[1], zorder=ZORDER, thickness=THICKNESS)
    r.append(parametric_plot((a*(1-t)+c, b*(1-t)+d), (t, 0, 1),
                              color = colors[2], zorder=ZORDER, thickness=THICKNESS
    r.append(parametric_plot((c*(1-t), d*(1-t)), (t, 0, 1),
                             color = colors[3], zorder=ZORDER, thickness=THICKNESS)
    # Dots make a cleaner join between edges
    r.append(circle((a, b), DOT_SIZE,
                    color = colors[0], zorder = 2*ZORDER, thickness = THICKNESS*1.2
    r.append(circle((a+c, b+d), DOT_SIZE,
                    color = colors[1], zorder = 2*ZORDER+1, thickness = THICKNESS*1
    r.append(circle((c, d), DOT_SIZE,
                    color = colors[2], zorder = ZORDER+1, thickness = THICKNESS*1.2
    r.append(circle((0, 0), DOT_SIZE,
                    color = colors[3], zorder = ZORDER+1, thickness = THICKNESS*1.2
    return r

def plot_square_action(a, b, c, d, show_unit_square = False):
    """Show the action of the matrix with entries a, b, c, d on half
    of the unit circle, as the circle and the output curve, broken into
    colors.
     a, b, c, d  reals  Entries are upper left, ur, ll, lr.
    """
    colors = ['red', 'orange', 'green', 'blue']
    G = Graphics()        # holds graph parts until they are shown
    if show_unit_square:
        for f_part in color_square_list(1,0,0,1,colors):
            G += f_part
    for g_part in color_square_list(a,b,c,d,colors):
        G += g_part
    p = plot(G)
    return p

plot.options['figsize'] = 2.5
plot.options['axes_pad'] = 0.05
plot.options['fontsize'] = 7
plot.options['dpi'] = 500
plot.options['aspect_ratio'] = 1
# plot.options['axes_range'] = (-4,4,-4,4)

# p = plot_square_action(1,1,0,1)
# p.set_axes_range(-4,4,-4,4)
# figure = p.matplotlib()
# print repr(figure.axes)
# show(figure, aspect_ratio=1)
# p.save("sageoutput/plot_action1.png")
```



Bibliography

Robert A. Beezer. Sage for Linear Algebra. http://linear.ups.edu/download/fcla-2.22-sage-4.7.1-preview.pdf, 2011.

Jim Hefferon. Linear Algebra. http://joshua.smcvt.edu/linearalgebra, 2012.

David Joyner and William Stein. Open source mathematical software. *Notices of the AMS*, page 1279, November 2007.

Sage Development Team. Sage tutorial 5.3. http://www.sagemath.org/pdf/SageTutorial.pdf, 2012a.

Sage Development Team. Sage reference manual 5.3. http://www.sagemath.org/pdf/reference.pdf, 2012b.